

***AutomaticDeployment* Aplikasi berbasis *Microservice* Pada  
Platform Kubernetes Dengan Metode *Pull Up Based***

**Proposal Tugas Akhir**

Diajukan Untuk Memenuhi  
Persyaratan Guna Meraih Gelar Sarjana  
Informatika Universitas Muhammadiyah Malang



**Muhammad Zein Ihza Fahrozi**  
**201810370311072**

**PROGRAM STUDI TEKNIK INFORMATIKA FAKULTAS TEKNIK**  
**UNIVERSITAS MUHAMMADIYAH MALANG**  
**2021**

## LEMBAR PERSETUJUAN

### ***Automatic Deployment* Aplikasi berbasis *Microservice* Pada Platform Kubernetes Dengan Metode *Pull Up Based***

**Muhammad Zein Ihza Fahrozi**  
**201810370311072**

Telah Direkomendasikan Untuk Diajukan Sebagai  
Judul Tugas Akhir Di  
Program Studi Informatika Universitas Muhammadiyah Malang

Malang, 2 Pebruari 2022  
Menyetujui,

Dosen I

Dosen II

Agus Eko Minarno, S.Kom., M.Kom.  
NIP. 108.1410.0540

Ilyas Nuryasin S.Kom, M.Kom.  
NIP. 108.1410.0561

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Sebuah perangkat lunak sebelum digunakan oleh pengguna secara luas perlu dilakukan proses deployment pada perangkat lunak tersebut. Deployment sebuah perangkat lunak dapat didefinisikan sebagai akuisisi dan eksekusi sebuah perangkat lunak. Proses ini biasanya dilakukan oleh seorang software deployer atau dalam bahasa yang sering digunakan belakangan ini seorang SRE (site reliability engineer)[1]. Maka dari itu dapat dikatakan deployment adalah aktivitas post-production sebuah perangkat lunak untuk digunakan oleh konsumen. Proses deployment sebuah perangkat lunak terdiri dari beberapa proses yang saling berhubungan seperti proses release sebuah perangkat lunak, instalasi perangkat lunak kedalam environment execution, dan aktivasi sebuah software[2].

Untuk mendeploy sebuah sistem perangkat lunak juga ada beberapa yang harus diperhatikan antara lain sub-komponen yang dibutuhkan atau package external, resource (hardware). Untuk melakukan deployment sub-komponen ini dibutuhkannya sebuah konfigurasi yang mendeskripsikan versi sub-komponen yang digunakan oleh perangkat lunak. Dengan bahasa modern sekarang konfigurasi sub-komponen yang digunakan oleh main aplikasi biasanya sudah otomatis terbuat contohnya antara lain adalah; go modules[3] dalam bahasa Golang, requirement file[4] dalam bahasa Python, atau file RubySpec[5] pada bahasa Ruby.

Terdapat beberapa karakteristik yang mendasar pada deployment sebuah perangkat lunak yang ditulis oleh Alan Dearle yaitu[6]; **Release** berupa jembatan antara proses deployment dengan proses development. yang meliputi semua operasi yang diperlukan untuk mempersiapkan sebuah sistem untuk di-transfer ke konsumen. Aktivitas release ini juga menentukan resource yang dibutuhkan oleh sebuah sistem perangkat lunak untuk dapat beroperasi pada environment nya. Setelah itu dilakukan packaging pada sistem perangkat lunak. Package tersebut harus mengandung komponen yang dibutuhkan oleh sistem, deskripsi sistem, dependencies pada komponen eksternal, prosedur deployment, dan semua informasi yang relevan dari sistem tersebut pada environment yang akan dijalankan. **Installation** diperlukan untuk persiapan melakukan activation. **Activation** adalah proses eksekusi sebuah perangkat lunak pada waktu tertentu, biasa menggunakan grafik antar muka ataupun

proses daemon. Updating adalah proses untuk mengganti bagian dari perangkat lunak yang terinstal dengan versi yang lebih baru. Selanjutnya yaitu **Undeployment** yaitu proses menghapus software yang terinstall dalam sebuah mesin ini dapat disebut dengan *deinstallation*.

Menurut Mockus dkk[7] kualitas deployment sebuah perangkat lunak masuk kedalam faktor utama dalam persepsi konsumen dalam hal kualitas sebuah perangkat lunak. Jansen dan Brinkkemper[8] juga mengatakan bahwa kelancaran sebuah deployment perangkat lunak adalah hal esensial untuk meningkatkan produk perangkat lunak sebuah perusahaan/organisasi. Tetapi terdapat beberapa tantangan yang dihadapi pada saat melakukan aktivitas deployem. Menurut Antonio Carzaniga[2] terdapat beberapa tantangan yang sering dihadapi pada saat melakukan deployment yaitu; **Mengganti** atau melakukan update sebuah sistem terhadap komponen yang sudah berjalan, **Dependencies** komponen antar satu sama lain, dan **Koordinasi** ketika melakukan update apakah akan mengganggu proses bisnis yang sedang berjalan atau tidak, dan juga **Mengatur** platform yang heterogen misalnya terhadap spesifik sistem operasi yang digunakan.

Seiring berjalannya waktu, sebuah aplikasi cenderung menjadi semakin kompleks[9, 10]. Dengan team pengembang dan aplikasi yang selalu bertumbuh dari sisi kompleksitas dan *maintainability* biasanya menyebabkan model pengembangan aplikasi menjadi susah untuk dikembangkan atau dapat dikatakan terdapat *bottleneck* sehingga aplikasi menjadi tidak efisien[11]. Dengan berkembangnya kompleksitas sebuah aplikasi diperlukan sebuah arsitektur yang bisa menyelesaikan hal itu, salah satu caranya yaitu menggunakan arsitektur *microservice*[9]. Dengan *microservice* aplikasi dibagi menjadi bagian-bagian kecil (unit) yang terpisah satu dengan lainnya[9]. Masing-masing bagian aplikasi ini dapat dijalankan dan dikembangkan secara independent (dari sisi developer)[12].

Saat ini aplikasi berbasis *microservice* menjadi pilihan sebagai arsitektur utama ketika membangun aplikasi yang scalable[13]. Aplikasi berbasis *microservice* ini dulunya dikembangkan dengan dengan menjalankan beberapa VM (virtual machine) yang saling berkomunikasi satu sama lain melalui REST/HTTP (Hypertext Transfer Protocol) ataupun RPC (*Remote Procedure Call*)[14]. Karena pada saat itu deployment *microservice* itu sendiri masih berbasis VM yang membutuhkan operasi manual dan juga biaya yang mahal, aplikasi berbasis *microservice* pun menjadi susah dan kompleks dari sisi biaya dan waktu yang dibutuhkan untuk dilakukan pengembangan dan juga *scaling*[14] .

Teknologi VM sendiri sudah perlahan ditinggalkan ketika merancang *microservice*. Dengan adanya teknologi *containerization* aplikasi dapat dibungkus agar dapat dijalankan dengan mudah dan efisien [14]. Menjalankan aplikasi dengan abstraksi yang diberikan oleh sebuah *container* juga membawa fleksibilitas dalam pengelolaannya. Salah satu manfaatnya adalah scaling aplikasi yang jauh lebih mudah, yaitu hanya dengan melakukan penyesuaian jumlah *container* yang dijalankan[15].

*Containerization*[16] ini sangat berdampak pada aspek infrastruktur dan *runtime* sebuah aplikasi. Tapi dengan adanya *container* diperlukan juga sebuah sistem yang melakukan orkestrasi secara otomatis pada *container* tersebut. Dengan adanya kubernetes kita dapat memanfaatkan fitur fitur yang diberikan oleh kubernetes antara lain fitur automatic scaling[17, 18]. Saat ini untuk melakukan deployment sebuah *container (service)* dilakukan secara manual dengan merubah file deployment. Dengan demikian, diperlukan cara otomatis untuk melakukan deployment *service* yang baru/diubah.

Dengan banyaknya persaingan dalam dunia perangkat lunak yang terjadi dalam waktu ini sebuah perusahaan atau developer memerlukan waktu yang cepat untuk melakukan deployment sebuah perangkat lunak. Terdapat beberapa macam workflow sebuah deployment perangkat lunak saat ini, tetapi yang industri saat ini lakukan adalah metode DevOps. Metode DevOps sendiri merupakan metode yang digunakan untuk mengembangkan perangkat lunak yang menjembatani antara dua team yang terisolasi dalam struktur organisasi, contohnya adalah team pengembang (Dev) dan team operasi (Ops)[19]. Konsep DevOps sendiri memungkinkan team developer dan operasi untuk membangun sebuah perangkat lunak yang dapat dijalankan secara otomatis dan secara berkala dengan menggunakan alat bantu DevOps. Tujuan utama dari itu adalah untuk meningkatkan kecepatan, reliabilitas, dan perangkat lunak yang lebih baik. Dalam DevOps sendiri terdapat beberapa sub-metode yang digunakan yaitu, *Continuous Integration (CI)*, *Continuous Delivery (CDE)*, dan *Continuous Deployment (CD)* [20].

Walaupun DevOps sendiri memiliki beberapa keunggulan, tetapi implementasi CI,CDE, dan CD bukanlah hal yang mudah. Pemilihan tools dan adaptasinya, adaptasi karyawan, dan miskonfigurasi yang biasa terjadi pada saat migrasi menggunakan metode DevOps.

Terdapat beberapa masalah yang sering terjadi pada saat menggunakan metode DevOps yaitu; Arsitektur[19], tools[21], metode baru [22, 23], Keamanan dari flow CI/CD[24], dan mekanisme rollback[25]. Pada penelitian yang dilakukan oleh Ramadoni dkk.[26] dilakukan analisis menggunakan metode GitOps yang dapat menyelesaikan permasalahan mekanisme rollback, dan keamanan flow CI/CD

Pada penelitian yang dilakukan oleh Ramadoni dkk.[26] digunakan metode pull-based untuk menyelesaikan permasalahan diatas. Pada penelitian tersebut tidak dijelaskan perbedaan kenapa harus menggunakan metode pull-based atau push-based dan juga tidak dijelaskan perbandingan tools yang digunakan sebagai operator GitOps yaitu ArgoCD pada sebuah cluster Kubernetes. Maka tujuan dari penelitian ini adalah untuk melakukan analisis perbedaan tools yang digunakan sebagai operator GitOps dan melakukan perbandingan metode pull-based dan push-based pada metode DevOps.

## **BAB II**

### **LANDASAN TEORI**

- 2.1 Tinjauan Studi**
- 2.2 Landasan Teori**
- 2.3 Pengembangan Sistem**
- 2.4 Perangkat Lunak Pendukung**

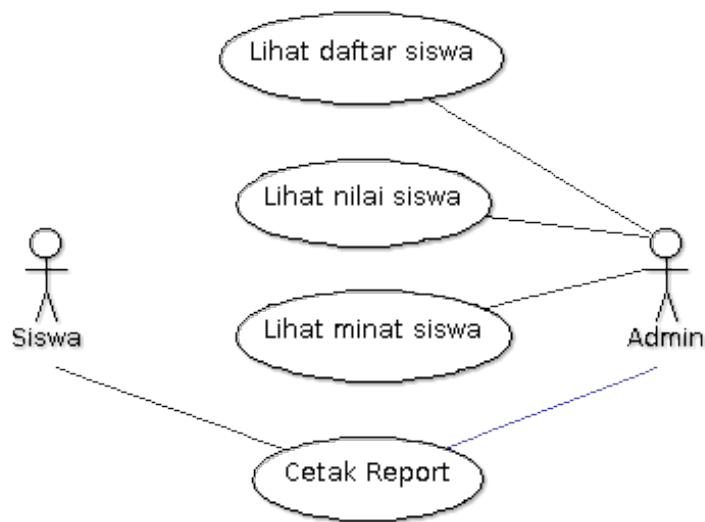
## BAB III

### ANALISA DAN PERANCANGAN SISTEM

#### 3.1 Analisa Sistem

##### 3.1.1 Analisa Sistem Saat Ini

Analisa sistem pendukung keputusan dalam penentuan penjurusan dibuat oleh peneliti dalam bentuk use case diagram yang mewakili secara sederhana dan bisa dijadikan sebagai bahan dalam evaluasi sistem yang berjalan, sehingga sistem dapat terlihat tanpa harus mengetahui secara detail prosedur yang berjalan.



**Gambar 3.1** Use Case Diagram Analisa Sistem Saat Ini



Dibawah ini merupakan deskripsi dari use case yang sedang berjalan:

1. *Admin* melihat daftar siswa.
2. *Admin* melihat nilai setiap siswa.
3. *Admin* melihat minat setiap siswa.
4. *Admin* mencetak hasil keputusan.
5. Siswa melihat laporan penjurusan yang telah dicetak oleh *admin*

### 3.1.2 Evaluasi Sistem Saat Ini

**Tabel 3.1** Permasalahan dan Solusinya

Masalah	Aktor	Solusi
1. Masalah masalah masalah Masalah masalah masalah Masalah masalah masalah Masalah masalah masalah. 2. Masalah masalah masalah Masalah masalah masalah Masalah masalah masalah Masalah masalah masalah. 3. Masalah masalah masalah Masalah masalah masalah Masalah masalah masalah Masalah masalah masalah.	1. Aktor 1 2. Aktor 2	1. Solusi solusi solusi Solusi solusi solusi Solusi solusi solusi Solusi solusi solusi Solusi solusi solusi. 2. Solusi solusi solusi Solusi solusi solusi Solusi solusi solusi Solusi solusi solusi Solusi solusi solusi. 3. Solusi solusi solusi Solusi solusi solusi Solusi solusi solusi Solusi solusi solusi.

### 3.1.3 Model yang Diusulkan

#### 3.1.4 Acitivity Diagram yang Diusulkan

#### 3.1.5 Perancangan Prosedur Sistem

##### 3.1.5.1 Use Case Diagram

##### 3.1.5.2 Activity Diagram

1. *Activity diagram* satu
  - a. Item 1.
  - b. Item 2.

2. Dua

##### 3.1.5.3 Class Diagram

##### 3.1.5.4 Sequence Diagram

#### 3.1.6 Perancangan Antarmuka (Interface)

## DAFTAR PUSTAKA

- [1] M. R. Lyu, “Software reliability engineering: A roadmap,” 2007, pp. 153–170.
- [2] A. Carzaniga, A. Fuggetta, R. S. Hall, D. Heimbigner, A. V. D. Hoek, and A. L. Wolf, “A characterization framework for software deployment technologies,” 1998.
- [3] “The go blog on go modules,” Jul 2021. [Online]. Available: <https://go.dev/blog/using-go-modules>
- [4] “How to install python packages with pip and requirements.txt.” [Online]. Available: <https://note.nkmk.me/en/python-pip-install-requirements/>
- [5] O. S. Software, “Ruby/spec: The ruby spec suite aka ruby/spec.” [Online]. Available: <https://github.com/ruby/spec>
- [6] A. Dearle, “Software deployment, past, present and future,” 2007.
- [7] A. Mockus and P. Zhang, “Predictors of customer perceived software quality,” 2005.
- [8] S. Jansen and S. Brinkkemper, “Definition and validation of the key process of release, delivery and deployment for product software vendors: turning the ugly duckling into a swan,” 2006. [Online]. Available: <http://www.swebok.org>
- [9] T. Llorido-Botran, J. Miguel-Alonso, and J. A. Lozano, “A review of auto-scaling techniques for elastic applications in cloud environments,” *Journal of Grid Computing*, vol. 12, pp. 559–592, 11 2014.
- [10] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 1st ed. O’Reilly Media, 2015.
- [11] Y. Yu, H. Silveira, and M. Sundaram, “A microservice based reference architecture model in the context of enterprise architecture.” IEEE, 10 2016, pp. 1856–1860.
- [12] Z. Xiao, I. Wijegunaratne, and X. Qiang, “Reflections on soa and microservices.” Institute of Electrical and Electronics Engineers Inc., 3 2017, pp. 60–67.
- [13] J. Q. Wu and T. Wang, “Research and application of soa and cloud computing model.” Institute of Electrical and Electronics Engineers Inc., 12 2014, pp. 294–299.

- [14] H. Khazaei, C. Barna, N. Beigi-Mohammadi, and M. Litoiu, "Efficiency analysis of provisioning microservices," 2016.
- [15] V. Singh and S. K. Peddoju, "Container-based microservice architecture for cloud applications." IEEE, 5 2017, pp. 847–852.
- [16] Davidbritch, "Containerized microservices - xamarin." [Online]. Available: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/containerized-microservices#:~:text=with%20client%20apps,-,Containerization,to%20a%20host%20operating%20system>.
- [17] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying microservice based applications with kubernetes: Experiments and lessons learned," vol. 2018-July. IEEE Computer Society, 9 2018, pp. 970–973.
- [18] "What is kubernetes?" Jul 2021. [Online]. Available: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>
- [19] R. Bolscher and M. Daneva, "Designing software architecture to support continuous delivery and devops: A systematic literature review." SciTePress, 2019, pp. 27–39.
- [20] G. Kim, K. Behr, and G. Spafford, *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win*, 1st ed. IT Revolution Press, 2013.
- [21] A. Proulx, F. Raymond, B. Roy, and F. Petrillo, "Problems and solutions of continuous deployment: A systematic review," 12 2018. [Online]. Available: <http://arxiv.org/abs/1812.08939>
- [22] M. K. A. Abbass, R. I. E. Osman, A. M. H. Mohammed, and M. W. A. Alshaikh, "Adopting continuous integration and continuous delivery for small teams." IEEE, 9 2019, pp. 1–4.
- [23] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of devops concepts and challenges," 11 2019.
- [24] M. Shahin, M. A. Babar, and L. Zhu, "Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices," pp. 3909–3943, 2017.
- [25] J. Fritzsche, J. Bogner, S. Wagner, and A. Zimmermann, "Microservices

migration in industry: Intentions, strategies, and challenges.” Institute of Electrical and Electronics Engineers Inc., 9 2019, pp. 481–490.

- [26] Ramadoni, E. Utami, and H. A. Fatta, “Analysis on the use of declarative and pull-based deployment models on gitops using argo cd.” Institute of Electrical and Electronics Engineers (IEEE), 10 2021, pp. 186–191.

# LAMPIRAN

## **Coding Login Aplikasi**